MASARYKOVA UNIVERZITA
FAKULTA INFORMATIKY


# Race Conditions in Message Sequence Charts

BACHELOR THESIS

**Petr Slovák**

Brno, spring 2008

## Declaration

Hereby I declare, that this paper is my original authorial work, which I have worked out by my own. All sources, references and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

**Advisor:** RNDr. Vojtěch Řehák, Ph.D.

## Acknowledgement

I'd like to express my deep gratitude to my supervisor, RNDr. Vojtěch Řehák, Ph.D., who has not only guided my efforts related to this thesis, but who has also taught me many valuable things about scientific work in general.

I would also like to thank Jan Fousek for his cooperation on the topic, patient proof reading as well as his numerous valuable comments.

## Abstract

Race condition problem is known to be decidable for Message Sequence Charts (MSC) but it becomes undecidable for High-level MSC. This thesis introduces a modification of the race condition problem, so called *trace race condition problem*, that remains decidable even for High-level MSC (with coregions).

In this thesis, we discuss differences between the *race condition* and the *trace race condition* problem and we argue that our concept is useful in model based design. We also propose an algorithm solving the *trace race condition* problem as well as show the problem itself to be NP-complete for HMSC with coregions.

## Keywords

Message Sequence Charts, High-level Message Sequence Charts, race conditions, trace race conditions

# Contents

## Chapter 1

# Introduction

As new systems get more and more complex, the design phase of the system development cycle is becoming even more important. The effect of a system design phase is, unfortunately, double edged: while a correct and efficient design can speed up the whole project considerably, an error in this phase can get dramatically more expensive and difficult to repair during subsequent phases. Therefore, the possibility to identify potentially problematic parts of the design while still designing can be very useful and cost-effective.

At the same time, design phase acts as a transformation between requirements placed upon the system and the system implementation itself. As requirements focus mainly on wanted behaviour of the system and are not, usually, created with the technical background in mind, this transformation might be very hard and error-prone.

Such situation is even more complex in the domain of distributed systems (e.g. communication protocols), where the requirements define the global behaviour (which might be sensitive to the current global state of the system), but individual components in the real implementation have only limited information about currect state of other processes.

It would be, therefore, preferable to be able to the system design on one side close to the intuitive description of the system (which leads to easier and less error-prone transformation), but at the same time keep the possibility of some form of formal system validation against the requirements.

Widely accepted solution is the use of formal languages, which, besides unambiguous specification/description of the system, might allow us to check some interesting properties of our design. One of the formalisms well suited for the problems encountered in the field of distributed systems is Message Sequence Charts (MSC) formalism.

ITU organisation introduced MSC in [16] as a formal language for specification and description of communication behaviour of system components and their environment. Emphasis is placed mainly upon message interchanges while some other implementation details (such as data storage

and computations in components) are abstracted away. In the visual representation of MSC, individual components of the system are represented as vertical lines called *instances* in the ITU specification and *processes* in most theoretically oriented papers, which is adopted also in this thesis. Messages are depicted as arrows between processes and intended order of the send and receive events is given by their location on the process line.

This formalism shares most of the nice properties mentioned in previous paragraphs: intuitive visual representation is used for description of individual behaviours of the system without implementation details, while the formal background allows the designers to formulate properties of interest and validate the system against them.

This bachelor thesis focuses mainly on one of important properties of systems described via MSC formalism called *race conditions*. As our main result, we propose a new notion of *trace race conditions* over an extension of MSCs called High-level Message Sequence Charts and analyze its language and algorithmic properties.

The structure of the thesis is therefore as follows: in the next chapter, we summarize previous work and compare our new approach with the former one. Chapter 3 then formally defines all used concepts. Chapter 4 contains the main result of this thesis – algorithm deciding *trace race conditions* in MSC-graphs together with its proof of correctness. In the following chapter, we discuss complexity of our algorithm. Chapter 6 focuses on other relevant work and possibilities for future extensions, while the last chapter concludes this thesis.

## Chapter 2

# (Trace) Race Conditions

## 2.1 Previous work

One of the first papers considering analysis of MSC for design inconsistencies was [3]. Here, Alur, Holzmann and Peled describe an important source of design errors which they call *race conditions*. Informally, a race condition occurs if two events on the same process are ordered in one way in the defining MSC, but could be ordered in the opposite order during an actual execution of the system. E.g. an order of two messages received from non-synchronised processes (see Figure 2.1) cannot be guarantied in an implementation. This inconsistency can lead to severe problems such as deadlocks or other unspecified behaviour of the system. In the same paper, authors propose a quadratic algorithm for finding race conditions in an MSC.

When the MSC formalism was introduced, only the power of individual MSCs was given to designers. Therefore, if designers wanted to specify behaviour of the whole system, they had to do so by specifying a (finite) collection of individual MSCs – each representing one possible execution. While race condition problem is still decidable in this setting (see [11]), the collections were enormous for complex systems and the MSC formalism has, at least in some cases, lost its clarity.

In 1996, ITU has extended the MSC formalism with *High-level Message Sequence Charts* (HMSC) [15]: these are finite state transition systems with states labelled by an MSC or an HMSC. Each accepting path (a run) in this transition system can be understood as an MSC specifying one possible behaviour of the system. Naturally, the MSC is a concatenation of MSCs labelling the states visited during the run. Therefore, HMSC can be used as a specification of the whole system (with possibly infinite number of different runs).

A natural question was, whether the concept of race conditions can be extended also to HMSCs. The first possible extension was proposed by Muscholl and Peled in [23] and the problem was proved to be undecid-

able for unrestricted version of HMSCs in the same paper. An alternative proof can be found in [1].

## 2.2 Race Conditions vs. Trace Race Conditions

Depending on the underlying semantics, an MSC may induce more than one possible execution – i.e., order of events on individual processes. For example, if we assume the processes to be non-synchronised, the system described by an MSC in Figure 2.1 may also induce behaviour shown in the Figure 2.2 (process C cannot know, whether process B has already received the message from process A or not).

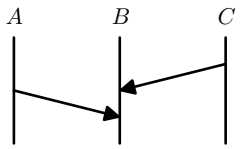


Figure 2.1: Race condition in MSC

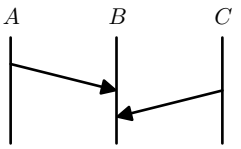


Figure 2.2: Race condition in MSC

Originally, MSCs were designed to describe examples of behaviour in a system and the system itself could be modelled only by sets of independent MSCs. Should the specification be complete, each induced execution of the system had to be included in the set. It was therefore natural to formulate the race condition problem for sets of MSCs as (informally said) a requirement that the specification should include all possible executions of any contained MSC.

Similar extension was originally used for HMSCs – an HMSC was considered correct (race free) if, for each valid run, all induced executions of this run were explicitly specified by some run/MSC of the HMSC. This is the intuition behind the original race condition problem, which was shown to be undecidable [23, 1].

These approaches describe one logical behaviour of a system eventually in two or more MSCs: each of them represents one potential ordering of independent events and the system is considered race free if all such possible orderings are defined in the system. E.g. the HMSC in Figure 2.3 is race free because it contains both orders of the receive events, on the other hand, one logical behaviour, stating that both $A$ and $C$ send a message to $B$, is split into two branches.

In this thesis, we propose a novel concept of race conditions. The core idea is to apprehend different MSCs created from runs in the HMSC as

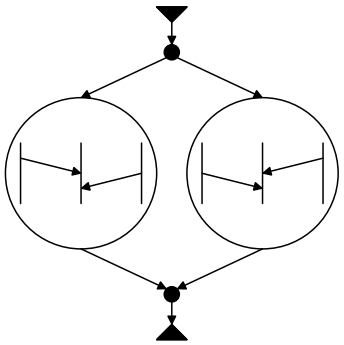

Figure 2.3: HMSC with a trace race condition

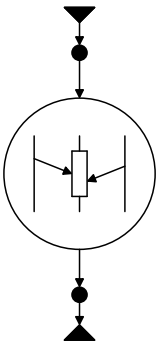

Figure 2.4: The coregion alternative

representations of logically distinct behaviours. Therefore, all possible ambiguities of a behaviour in a system must be dealt with within the run that specifies it. The example in Figure 2.3 illustrates this difference between the two race condition concepts – the system in Figure 2.3 does not contain a race in the original race condition problem, but does contain a race in the trace race condition problem.

If we want to describe a logical behaviour (e.g., the one in Figure 2.3) in one MSC, simple MSC might not be powerful enough. Fortunately, MSC formalism includes the notion of a *coregion*. This concept allows us to mark several consequent events on a process as mutually independent – they can happen in any possible order. Therefore, usage of a coregion in Figure 2.4 can be understood as a compact way of describing the design shown in Figure 2.3.

As it was already pointed out in [11], coregions allow designers to express the designed system in a succinct way; thus making the model of the system less cumbersome and easier to understand. This approach was also recommended by the ANF DATA (Siemens AG) company, as keeping each logical behaviour separate might encourage a more clean way of system design.

To sum up, instead of HMSC with undecidable race condition problem, we suggest using HMSC with coregions, specifying all possible ambiguities within each run and checking races by our algorithm for the trace race condition problem. Due to potentially infinite number of possible MSCs included in the system, it is, however, impossible to solve the *trace race condition* problem just by a trivial modification of the algorithm proposed in [3]. Therefore, a more sophisticated approach is needed and is described in the following sections.

## Chapter 3

# Preliminaries

In this section, we recall some preliminary definitions to unify our notation.

## 3.1 MSC with Coregions and Connections

A coregion is introduced into the MSC formalism for the specification of unordered events on a process. We use a similar definition to the one proposed in [11] as coregion of type 2. Additionally, we restrict the coregions to be pairwise distinct (to comply with ITU specification). In contrast to [11], we also put to use *connections* within a coregion as a further synchronization construct: this is a subclass of the *general order relation* introduced in the ITU specification of MSC.

**Definition 3.1.1** *An* MSC *(with coregions and connections) is defined as a tuple* $(E, <, \mathcal{P}, \mathcal{T}, P, \mathcal{M}, \mathcal{C}, \mathcal{G})$ *where*

- $E$ *is a set of* events*;*
- $<$ *is a partial ordering on* $E$ *called* visual order*;*
- $\mathcal{P}$ *is a finite set of* processes*;*
- $\mathcal{T} : E \to \{s, r\}$ *is a labelling function dividing events into two types* send *and* receive*;*
- $P : E \to \mathcal{P}$ *is a mapping that associates each event with a process;*
- $\mathcal{M} \subseteq (\mathcal{T}^{-1}(s) \times \mathcal{T}^{-1}(r))$ *is a bijective mapping, relating every send with a unique receive, such that for any* $e, f \in \mathcal{M}$ *we have* $P(e) \neq P(f)$ *– a process cannot send messages to itself;*
- $\mathcal{C}$ *is a set of pairwise disjoint* coregions *where a coregion (say* $C \in \mathcal{C}$*) is defined as a subset of events on some process, i.e.* $C \subseteq P^{-1}(p)$ *where* $p \in \mathcal{P}$*;*

- $\mathcal{G} \subseteq \bigcup_{C \in \mathcal{C}} C \times C$ *is a partial ordering on events within coregions and is called* connections.

Visual order $<$ *is defined as the reflexive and transitive closure of*

$$\mathcal{M} \cup \mathcal{G} \cup \bigcup_{p \in \mathcal{P}} <_p \text{ where } <_p = (\prec_p \setminus \{(e_x, e_y) \mid e_x, e_y \in C \ \wedge \ C \in \mathcal{C}\})$$

*where* $\prec_p$ *is a total order on* $P^{-1}(p)$ *such that each coregion associated with* $p$ *is a set of consecutive events wrt* $\prec_p$*. In other words,* $<_p$ *is a total ordering of events outside of the coregions and of the coregions as whole units; each two events within the coregions are pairwise unordered.*

*Visual order* specifies ordering for each two events on the same process (that do not belong to the same coregion). As was already mentioned in previous sections, these two events might be ordered differently in an actual execution due to some underlying behaviour of the system (e.g. fifo/non-fifo, synchronous/asynchronous delivery of messages). It is therefore useful to define so-called *causal order*, that orders individual events with respect to possible executions of the system.

As these executions are dependent on underlying semantics, definition of the *causal order* must be altered accordingly. In this thesis, we will use the fifo semantics only. Nevertheless, we suggest our algorithm to work (with slight modifications) also for other semantic models such as non-fifo, global fifo, etc.

**Definition 3.1.2** *Given an MSC* $M = (E, <, \mathcal{P}, \mathcal{T}, P, \mathcal{M}, \mathcal{C})$*, we define a* causal ordering $\ll$ *as the least partial ordering on* $E$ *such that* $e \ll f$*, if at least one of the following items holds:*

- $(e, f) \in \mathcal{M}$ – send and receive of a message are ordered
- $P(e) = P(f)$ *and* $e < f$ *and* $\mathcal{T}(f) = s$ – any send event is delayed until all previous events took place
- $P(e) = P(f)$ *and* $\exists e', f' \in E$ *such that* $e' \ll f'$, $P(e') = P(f')$, $(e', e) \in \mathcal{M}$ *and* $(f', f) \in \mathcal{M}$ – fifo condition

## 3.2 MSC-graphs

In accordance with previous work on HMSCs, e.g. [2, 23], we will use a semantically equivalent notion of *MSC-graphs* instead of HMSC.

Informally, an MSC-graph $G$ is a finite graph with *initial* and *terminal* vertex, set of MSCs and a labelling function, that labels one MSC from the set to each vertex of $G$. Each finite path starting in the initial vertex and ending at the terminal vertex is considered to be a *run* of the MSC-graph $G$ and, as such, represents one execution of the system modelled by $G$.

To formalise this definition, we need to define a concatenation operation on MSCs. Intuitively, concatenation of two MSCs $M_1$ and $M_2$ is done by gluing the corresponding process lines together with the MSC $M_2$ drawn beneath $M_1$.

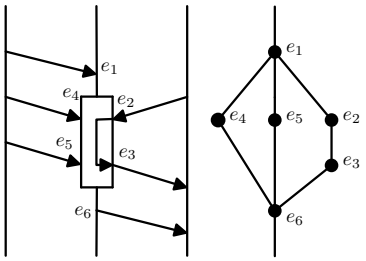


Figure 3.1: Coregion and its partial order

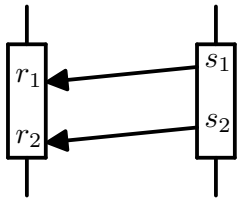


Figure 3.2: Non-trivial coregions scenario

**Definition 3.2.1** *Given two MSCs* $M_1 = (E_1, <_1, \mathcal{P}, \mathcal{T}_1, P_1, \mathcal{M}_1, \mathcal{C}_1)$ *and* $M_2 = (E_2, <_2, \mathcal{P}, \mathcal{T}_2, P_2, \mathcal{M}_2, \mathcal{C}_2)$ *over a common process set* $\mathcal{P}$ *and with* $E_1 \cap E_2 = \emptyset$*, let the concatenation of* $M_1$ *and* $M_2$ *be the MSC* $M_1 \cdot M_2 = (E_1 \cup E_2, <, \mathcal{P}, \mathcal{T}_1 \cup \mathcal{T}_2, P_1 \cup P_2, \mathcal{M}_1 \cup \mathcal{M}_2, \mathcal{C}_1 \cup \mathcal{C}_2)$ *where* $< = <_1 \cup <_2 \cup \bigcup_{p \in \mathcal{P}} (P_1^{-1}(p) \times P_2^{-1}(p))$.

**Definition 3.2.2** *An* MSC-graph $G = (S, \rightarrow, s_0, s_f, L, \mathcal{L})$ *consists of*

- *a finite set of states* $S$*,*
- *a transition relation* $\rightarrow \subseteq S \times S$*,*
- *an initial state* $s_0 \in S$*,*
- *a final state* $s_f \in S$*,*
- *a finite set of MSCs* $\mathcal{L}$*, and*
- *a mapping* $L$*, assigning to each state* $s \in S$ *an MSC* $L(s)$ *over set of MSCs* $\mathcal{L}$*.*

*A sequence of states* $s_1 s_2 \cdots s_k$ *is a* path*, if* $(s_i, s_{i+1}) \in \rightarrow$ *for every* $1 \leq i < k$*. A path is a* run*, if* $s_1 = s_0$ *and* $s_k = s_f$*.*

## 3.3 Race Conditions

Race conditions are usually defined by comparing linearizations of the appropriate MSC or MSC-graph wrt $<$ and $\ll$ relations (see [3, 11, 23]). The definition of linearization is quite simple for an MSC (resp. MSC-graphs) without coregions.

Unfortunately, situation is more complex when coregions are used as for some MSCs, the corresponding set of linearizations cannot be described by a single partial order ([11]). For example, consider a situation shown in Figure 3.2. In this case, any partial order containing the correct set of linearizations has to contain an incorrect (wrt fifo rule) linearization $(s_1, s_2, r_2, r_1)$ as well. For more details, see Example 1 in [11].

We must, therefore, use a slightly extended definition of linearization.

**Definition 3.3.1** *A* linearization *of an MSC $M = (E, <, \mathcal{P}, \mathcal{T}, P, \mathcal{M}, \mathcal{C}, \mathcal{G})$ according to a relation $\theta \in \{<, \ll\}$ is a* **total** *order $\sqsubseteq$ on $E$ such that $\theta \subseteq \sqsubseteq$ and, to satisfy the fifo condition, $e \sqsubseteq f$ if $P(e) = P(f)$ and $\exists e', f' \in E$ such that $e' \sqsubseteq f'$, $P(e') = P(f')$, $(e', e) \in \mathcal{M}$ and $(f', f) \in \mathcal{M}$. For an MSC $M$ we define $Lin_\theta(M)$ to be the set of all linearizations of $M$ according to $\theta$.*

Intuitively, this definition can be extended to MSC-graphs as follows.

**Definition 3.3.2** *For an MSC-graph, we define it's linearization as follows: $Lin_\theta(G) = \{l \mid l \in Lin_\theta(w),$ where $w$ is a run of $G\}$.*

Now we are ready to define the original race condition problem for MSC-graphs as defined in [23] as well as our *trace race condition* problem.

**Definition 3.3.3** *The original* **race condition** *problem is defined as follows: An MSC-graph G is said to contain a race if and only if $Lin_<(G) \neq Lin_\ll(G)$.*

**Definition 3.3.4** *The* **trace race condition** *problem is defined as follows: An MSC-graph G is said to contain a race if and only if there exist a run $w$ of $G$ such that $Lin_<(w) \neq Lin_\ll(w)$.*

In the rest of this thesis, we will use the phrase "a race between two events $e_1, e_2$" as a shorthand notation of the fact, that either $e_1 \ll e_2$ and $e_1 \not< e_2$, or $e_1 \not\ll e_2$ and $e_1 < e_2$.

**Remark 3.3.1** *Due to the introduction of coregions,* $\ll \subseteq <$ *no longer holds. Therefore, we might get a new type of race conditions, where* $e \ll f$ *but not* $e < f$ *(see events* $e_4$ *and* $e_5$ *in Figure 3.1). As these race conditions occur only inside individual MSCs, they are easy to deal with – we might either automatically solve them by an addition of a* connection*, or consider them equivalent to "standard" race conditions.*

Chapter 4

# Main Result

In this chapter, we show an algorithm solving the *trace race condition* problem for MSC-graphs (with coregions and connections). Let us first state some observations.

## 4.1 Initial Observations

Races occurring inside individual MSCs are easy to check by the algorithm proposed in [3]. The interesting part is checking races that are created by concatenating several MSCs. Therefore, we first create an algorithm deciding existence of a race condition in a concatenation of two MSC and use it later as a key part of the main algorithm for MSC-graphs.

Lemma 4.1.1 shows us, that to solve a race condition problem in a concatenation of two race free MSCs, only the minimal and maximal events of the corresponding MSCs need to be checked.

**Lemma 4.1.1** *Let us have two MSC's $M_1$, $M_2$ without race conditions and their concatenation $M_1 \cdot M_2$. Let there be a race in $M_1 \cdot M_2$ between events $e_1$ and $e_2$ such that $e_1$ is in $M_1$ and $e_2$ is in $M_2$. From the definition of race condition, both events are on the same process, let's say $p$. Then, maximal event on process $p$ in $M_1$ and minimal event on the same process in $M_2$ are also in a race.*

**sketch** The proof of the lemma directly follows from transitivity of *causal order* and the fact, that $M_1$ and $M_2$ are without race conditions. □

The next definition introduces three auxiliary functions to clear the notation in following proofs. The `Parent` function assigns the corresponding sending process to each receive event. $MaxP$ and $MinP$ functions return, for any MSC M, a set of events that are maximal, resp. minimal on their processes. Let us remark, that due to the addition of coregions, there is a possibility of more than one maximal, resp. minimal event on some process.

**Definition 4.1.1** *Let $M = (E, <, \mathcal{P}, \mathcal{T}, P, \mathcal{M}, \mathcal{C}, \mathcal{G})$ be an MSC. Let* `Parent` *be a function defined on receive events of $M$ such that* $\texttt{Parent}(e) = P(f)$ *where $(f, e) \in \mathcal{M}$. Also, define $MaxP(M)$ as a set $\{e \in E \mid \nexists e' \in E \,.\, P(e') = P(e) \wedge e \ll e'\}$. Analogically, $MinP(M) = \{e \in E \mid \nexists e' \in E \,.\, P(e') = P(e) \wedge e' \ll e\}$.*

We know from Lemma 4.1.1 that only the "border" events of two concatenated MSCs need to be checked. The following lemma states the exact conditions, under which two such events are related by the $\ll$ relation.

**Lemma 4.1.2** *Let us have a concatenation $M \cdot M'$ of two MSCs $M$ and $M'$ and two events $A \in MaxP(M)$ and $B \in MinP(M')$ such that $P(A) = P(B)$. Then*

*$A \ll B$ iff*

$$\begin{aligned} & \mathcal{T}(B) = s \\ \textit{or} \quad & \mathcal{T}(A) = \mathcal{T}(B) = r \ \wedge \ \texttt{Parent}(B) = \texttt{Parent}(A) \\ \textit{or} \quad & \exists\, C \in MaxP(M) \,.\, A \ll C \wedge \\ & \quad \exists\, D \in MinP(M') \,.\, D \ll B \ \wedge \ P(C) = P(D). \end{aligned}$$

**Proof** In the proof we discuss following cases:

- Let $B$ be a sent event, i.e. $\mathcal{T}(B) = s$, then $A < B$ due to the definition of concatenation, and so $A \ll B$.
- Let $B$ be a receive event, i.e. $\mathcal{T}(B) = r$.
  - Let $A$ be a receive event, i.e. $\mathcal{T}(A) = r$, then there are two events $C$ and $D$ such that $(C, A) \in \mathcal{M}$ and $(D, B) \in \mathcal{M}'$. It follows from the fifo condition that $A \ll B$ if $P(D) = P(C)$, i.e. $P(D) = \texttt{Parent}(A)$. Otherwise, due to the maximality of $A$ and the minimality of $B$, there is no subsequent communication from $P(C)$ to $P(A)$. Hence, nothing can force all other communication ending in $B$ to take longer time than the delivery ending in $A$.

    Therefore, $A \ll B$ if and only if $\texttt{Parent}(B) = \texttt{Parent}(A)$.
  - Let $A$ be a sent event, i.e. $\mathcal{T}(A) = s$. In what follows we show that $A \ll B$ if and only if there are two events $C \in MaxP(M)$ and $D \in MinP(M')$ such that $A \ll C$, $D \ll B$ and $P(C) = P(D)$. We discuss both implications:
    - Assume there are the events $C$ and $D$ and $\mathcal{T}(D) = s$. It follows directly from the definition of concatenation that $C <$

$D$. Definition of causal ordering then implies $C \ll D$, and $A \ll B$ holds.

On the other hand, let us assume there are the events $C$ and $D$ and $\mathcal{T}(D) = r$. Due to $D \ll B$ and minimality of $B$, we get that $P(D) \neq P(B)$. Therefore, there is a sent event $E$ on the same process as $D$ such that $D < E \ll B$. It follows from definition of concatenation that $C < E \ll B$ and the definition of causal ordering implies $A \ll B$.

* Finally, we assume that all events $C \in MaxP(M)$ and $D \in MinP(M')$ such that $A \ll C$ and $D \ll B$ are on different processes, i.e. $P(C) \neq P(D)$. Therefore, there is no communication from $A$ to any of the processes effecting the event $B$, i.e. $A$ and $B$ are in race.

Let us note that there is no event $C \in MaxP(M)$ such that $A \ll C$, if $\mathcal{T}(A) \neq s$. It holds similarly for the event $D$, if $\mathcal{T}(B) \neq r$.

□

The following definition introduces a *footprint* of a given MSC $M$ as a part of information that is, due to Lemma 4.1.3, sufficient for deciding existence of race conditions in MSC $M \cdot M'$, where $M'$ is an arbitrary MSC.

**Definition 4.1.2** *Let $M$ be an MSC. A* footprint *of $M$ is then defined as a tuple $(MaxP(M), \ll, \mathcal{P}, \mathcal{T}, P, \texttt{Parent})$ where*

- *$MaxP(M)$ is the set of maximal events of $M$,*
- *$\mathcal{P}$ is the set of processes of $M$,*
- *$\ll$ is the causal ordering of $M$ restricted onto $MaxP(M)$, and*
- *$\mathcal{T}$, $P$, $\texttt{Parent}$ are the functions of $M$ restricted onto $MaxP(M)$.*

The following is a direct consequence of Lemma 4.1.1 and Lemma 4.1.2.

**Lemma 4.1.3** *Let us have two race free MSCs $M$ and $M'$. We can decide whether there is a race in the concatenation of $M$ and $M'$ even if we know only the footprint of $M$ instead the whole of $M$. Also, for any two paths $p, p'$ with the same footprint $f$ and an MSC $X$ the following holds:*

- *$L(p) \cdot X$ is race free iff $L(p') \cdot X$ is race free*
- *$L(p) \cdot X$ and $L(p') \cdot X$ have the same footprint*

## 4.2 First Algorithm – Race in a Concatenation of Two MSCs

We are now ready to give the preliminary algorithm deciding existence of a race condition in a concatenation of two MSC.

**Algorithm 1 – Concatenation of Two MSCs**

**Input:** A footprint $(E_f, \ll_f, \mathcal{P}, \mathcal{T}_f, P_f, \texttt{Parent}_f)$ of a race free MSC $M_1$ and a race free MSC $M_2 = (E, <, \mathcal{P}, \mathcal{T}, P, \mathcal{M}, \mathcal{C}, \mathcal{G})$
**Output:** "race" if a race occurs in $M_1 \cdot M_2$, "no race" otherwise
**Body:**

**for every** $B \in MinP(M_2)$ such that $\mathcal{T}(B) = r$ **do**

**for every** $A \in E_f$ such that $P_f(A) = P(B)$ **do**

**if** $\mathcal{T}_f(A) = r$ and $\texttt{Parent}_f(A) \neq \texttt{Parent}(B)$ **then return** "race" **fi**

**if** $\mathcal{T}_f(A) = s$ and
$\{P_f(C) \mid C \in E_f \wedge A \ll_f C\} \cap \{P(D) \mid D \in MinP(M_2) \wedge D \ll B\} = \emptyset$
**then return** "race"

**fi**

**od**

**od**

**return** "no race"

**Correctness:** Follows from Lemma 4.1.1 and Lemma 4.1.2.

## 4.3 Second Algorithm – Deciding Trace Race in MSC-graphs

At this moment, if we have a *footprint* of an MSC $M$ and we know it to be race free, we can, for arbitrary MSC $M'$, decide existence of a race condition in the MSC $M \cdot M'$ as well. The following lemma states that we can also calculate a *footprint* of this concatenation.

**Lemma 4.3.1** *Let us have a footprint* $(E_f, \ll_f, \mathcal{P}, \mathcal{T}_f, P_f, \texttt{Parent}_f)$ *of a race free MSC* $M_1$ *and a race free MSC* $M_2 = (E, <, \mathcal{P}, \mathcal{T}, P, \mathcal{M}, \mathcal{C}, \mathcal{G})$ *such that* $M_1 \cdot M_2$ *is race free. Then we can, for the the concatenation* $M_1 \cdot M_2$*, compute a footprint* $(E_F, \ll_F, \mathcal{P}, \mathcal{T}_F, P_F, \texttt{Parent}_F)$ *corresponding to* $M_1 \cdot M_2$*.*

**Proof** The footprint of $M_1 \cdot M_2$ is a footprint of $M_2$ extended with all events $A \in E_f$ such that $P^{-1}(P_f(A)) = \emptyset$, i.e. in $M_2$, there is no event on the process $P_f(A)$. The function values of $\mathcal{T}_F(A), P_F(A), \texttt{Parent}_F(A)$ are equal to the ones of the footprint of $M_1$. The causal order $\ll_F$ on the event $A$ is defined as follows. For every $B \in E_F \cap E_f$ it holds that $A \ll_F B \iff A \ll_f B$. For every $B \in E_F \cap E$ it holds that $A \ll_F B \iff \{P_f(C) \mid C \in E_f \wedge A \ll_f C\} \cap \{P(D) \mid D = B \vee (D \in MinP(M_2) \wedge D \ll B)\} \neq \emptyset$. □

Using these results, we are finally able to construct the algorithm for deciding *trace race conditions* in a given MSC-graph $G = (S, \rightarrow, s_0, s_f, L, \mathcal{L})$.

Intuitively, the algorithm searches $G$ starting from state $s_0$ and remembers all found paths in form of pairs $(s, f)$, where $f$ is a footprint of some path $p = s_0 \dots s_n$ and $s$ is the last state of the path. When the tuple $(s, f)$ is processed, the algorithm inspects, for all steps $s \rightarrow s'$, if the path $s_0 \dots ss'$ is still without race conditions (using Algorithm 1).

In case of positive answer, we have found a race and the algorithm is terminated. Otherwise, the algorithm calculates the footprint $f'$ of the concatenation and examines, if a path with a footprint corresponding to $f' \cdot s'$ has been already processed. If yes, continuation of this path is unnecessary due to Lemma 4.1.3. In the opposite case, all possible extensions of $f' \cdot s'$ are added for processing.

**Algorithm 2 – Race in MSC-graph**

**Input:** An MSC-graph $G = (S, \rightarrow, s_0, s_f, L, \mathcal{L})$
**Output:** "race" if a race occurs in some run of the MSC-graph, "no race" otherwise
**Precomputation:**
Denote $S_{bad}$ the set of all states in $S$, for which does not exist a path ending in $s_f$ and remove them from $G$. Next, check if all MSCs in $L(S \setminus S_{bad})$ are race free.

**Body:**

$TODO := \{(s_0, footprint(L(s_0))\}$

**while** $TODO \neq \emptyset$ **do**

take $(s, f)$ from $TODO$

add $(s, f)$ to $DONE$

**for every** $s'$ such that $(s, s') \in \rightarrow$ **do**

**if** $f \cdot L(s')$ has a race **then** return "race"

$f' := footprint(f \cdot L(s'))$

**if** $(s', f') \notin TODO \cup DONE$ **then** add $(s', f')$ to $TODO$

**od**

**od**

**return** "no race"

**Correctness:** As each state $s$ can be associated with only a finite number of possible footprints the Algorithm 2 ends. Lemma 4.1.3 and correctness of Algorithm 1 give us correctness of Algorithm 2.

**Theorem 4.3.2** *The trace race condition problem for MSC-graphs/HMSCs is decidable.*

## Chapter 5

# Complexity Boundary

## 5.1 Trace Race in NP

In this section, we show that the trace race condition problem for MSC-graphs (with coregions and connections) is NP-complete.

**Lemma 5.1.1** *The trace race condition problem for MSC-graphs is in* NP.

**Proof** To prove the lemma, we discuss complexity of Algorithm 2.

Due to [3], it is known that the race condition problem for MSC without coregions is in P. A straightforward modification of the algorithm can solve the problem for MSC (with coregions and connections) in NP. Therefore, the precomputation of Algorithm 2 can be solved in NP.

It remains to show that the body of Algorithm 2 can be solved is in NP, too. The crucial point is to show that every reachable pair $(s, f)$ of $TODO$ can be reached in a run of a length polynomial to the size of the input MSC-graph $G = (S, \rightarrow, s_0, s_f, L, \mathcal{L})$. We prove this by an inspection of those parts of the run that are crutial for creation of the footprint $f$.

Given a reachable pair $(s_n, f)$, let us have a path $s_0 s_1 \cdots s_n$ such that $f = (E_f, \ll_f, \mathcal{P}, \mathcal{T}_f, P_f, \texttt{Parent}_f)$ is a footprint of $s_0 s_1 \cdots s_n$. It holds for every $p \in \mathcal{P}$ that all events of $P_f^{-1}(p)$ are introduced in the same state of the run. Hence, there are at most $|\mathcal{P}|$ states which include in their MSCs the events of $E_f$.

As a causal ordering $\ll_f$ is a part of the footprint $f$, we have to inspect all events inducing the causal relations between pairs of events of $E_f$. If both events of the inspected pair are on the same process, they are in the same MSC, say $L(s_i)$, and their causal order can be checked in $O(|L(s_i)|)$ time. If the events of $E_f$, say $A$ and $B$, are on different processes, we have to inspect a path of actions inducing the causal order of $A$ and $B$. By an action, we mean a pair of send event and receive event forming a message. $|\mathcal{P}|$ is the maximal number of actions that are necessary for inducing the causal order of $A$ and $B$ as each action enters a new process. Each of these

actions can be in an individual MSC – to reach a state with some particular action, we have to visit at most $|S|$ states. Therefore, to induce a causal order of two events of $E_f$, we need to visit at most $|S| \cdot |\mathcal{P}|$ states. As there are $|E_f|^2$ pairs of events, we need a path of at most $|E_f|^2 \cdot |S| \cdot |\mathcal{P}|$ states. We can conclude that every reachable pair $(s, f)$ can be reached in a run which length is $O(|G|^4)$. Therefore, Algorithm 2 solves the trace race condition problem for MSC-graphs in NP. □

## 5.2 NP-hardness of the Problem

**Lemma 5.2.1** *The trace race condition problem for MSC-graphs is* NP*-hard.*

**Proof** We prove this lemma by a reduction of SAT problem, which is known to be NP-complete [10], onto the trace race condition problem for MSC-graphs with coregions.

First, let us recall the definition of the SAT problem.

**Instance:** A set $U$ of variables and a collection $C$ of clauses over $U$.
**Question:** Is there a satisfying truth assignment for $C$?

Let $U = \{x_1, x_2, \cdots, x_n\}$ be a set of variables and $C = \{c_1, c_2, \cdots, c_m\}$ be a collection of clauses making up an arbitrary instance of the SAT problem. We shall construct an MSC-graph $G$ such that there is a trace race in $G$ if and only if $C$ is satisfiable.

Let $G$ be $(S, \rightarrow, s_0, s_f, L, \mathcal{L})$ where

- $S = \{s_0, t_0\} \cup \{s_i, t_{ij} \mid 1 \leq i \leq m \wedge 1 \leq j \leq |c_i|\} \cup$
  $\{s_{m+k}, v_k, nv_k \mid 1 \leq k \leq n\} \cup \{s_{m+n+1}, t_f, s_f\}$
- $\rightarrow = \{(s_0, t_0), (t_0, s_1)\} \cup \{(s_i, t_{ij}), (t_{ij}, s_{i+1}) \mid 1 \leq i \leq m \wedge 1 \leq j \leq |c_i|\} \cup$
  $\{(s_{m+k}, v_k), (s_{m+k}, nv_k), (v_k, s_{m+k+1}), (nv_k, s_{m+k+1}), \mid 1 \leq k \leq n\} \cup$
  $\{(s_{m+n+1}, t_f), (t_f, s_f)\}$,
- the finite set of MSCs $\mathcal{L}$ is induced by the following mapping and consists of MSCs expressing communication between processes $\{master, solver, x_k, \overline{x_k}, k, sync \mid 1 \leq k \leq n\}$
- the mapping $L$ assigns MSCs to states of $S$ as follows:
  - $L(s_i)$ is an empty MSC for every $0 \leq i \leq m + n + 1$ and $i = f$;
  - in $L(t_0)$, process $master$ sends a message to process $solver$;

- in $L(t_{ij})$, process $solver$ sends a message to the process labelled by the $j$-th literal of the clause $c_i$;
- in $L(v_k)$, process $x_i$ sends a message to the process labelled by $i$;
- in $L(nv_k)$, process $\overline{x_i}$ sends a message to the process labelled by $i$;
- in $L(t_f)$, processes $1, 2, \cdots, n$ send messages to process $sync$ and receive events of all this messages are in a coregion; after receiving all of this $n$ messages, process $sync$ sends a message to process $master$.

See Figure 5.1 for an example transformation of SAT instance given by the sets $U = \{x_1, x_2, x_3\}$ and $C = \{\{x_1, \overline{x_2}, x_3\}, \{x_1, \overline{x_3}\}\}$.

To prove that this is indeed a reduction, we must show that there is a trace race in $G$ if and only if $C$ is satisfiable.

Let $A$ be the send event of $L(t_0)$ and $B$ be the receive event on $master$ in $L(t_f)$. We show that $C$ is satisfiable if and only if there is a run of $G$, in which $A$ and $B$ are in race.

Suppose first that $\alpha : U \to \{T, F\}$ is a truth assignment satisfying $C$. Therefore, in $G$, there is a path from $s_0$ to $s_{m+1}$ such that, for every $1 \leq k \leq n$, $solver$ do not send any message to $x_k$ if $\alpha(x_k) = F$ (resp. to $\overline{x_k}$ if $\alpha(x_k) = T$). We extend this path to a run going through states $v_k$ if $\alpha(x_k) = T$ (resp. through $nv_k$ if $\alpha(x_k) = F$). In the MSC inducted by this run, for every $1 \leq k \leq n$ the send events of $v_k$ or $nv_k$ are the first events on these processes and so, they are not causally ordered with $A$. Therefore, neither $B$ is causally order with $A$ and they are in a race.

Let us assume that there is a run with a trace race in $G$. Quick inspection of the construction shows that only event $A$ and event $B$ can be in race. Moreover, all events in the MSCs before the state $s_{m+1}$ are causally dependent successors of $A$ and all events in the MSCs after $s_{m+1}$ are causally dependent predecessors of $B$. Therefore, in the race-inducing run, the events after the state $s_{m+1}$ have to be causally independent with $A$ and all events in the MSCs before the state $s_{m+1}$. In other words, the events "before $s_{m+1}$" has to be executed on different processes than the ones "after $s_{m+1}$", except of $A$ and $B$. This implies that there is an assignment $\beta : U \to \{T, F\}$ such that the events "before $s_{m+1}$" do not affect process $x_k$ if $\beta(x_K) = F$ (resp. $\overline{x_k}$ if $\beta(x_K) = T$). Therefore, $\beta$ is a truth assignment satisfying $C$. □

The following theorem directly follows from Lemma 5.1.1 and Lemma 5.2.1.

**Theorem 5.2.2** *The trace race condition problem for MSC-graphs/HMSCs (with coregions and connections) is* NP*-complete.*

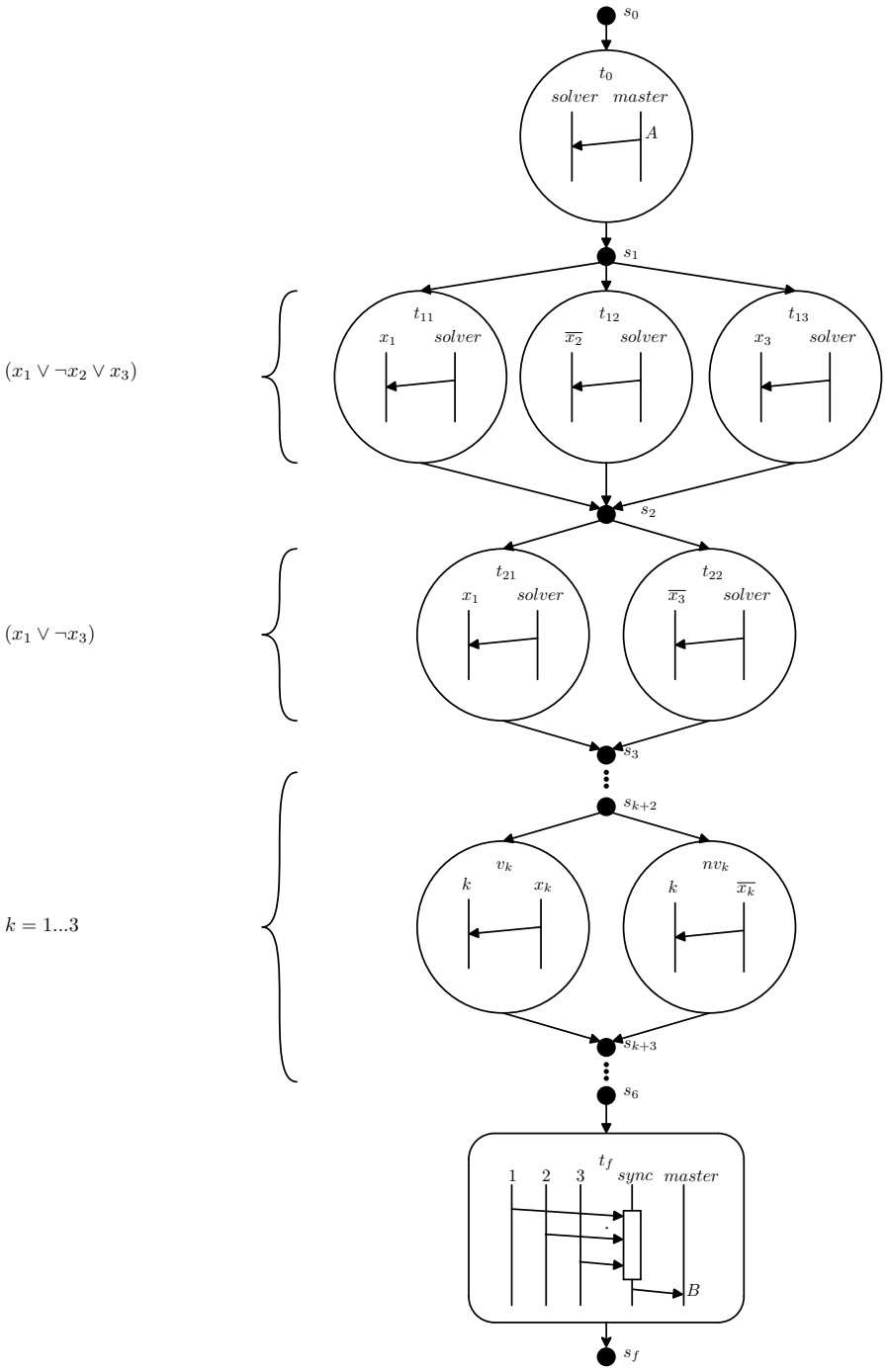


Figure 5.1: SAT instance reduction example

## Chapter 6

# Related Topics and Future Work

## 6.1 Related Topics

All papers directly considering algorithms for detection of race conditions either in MSCs or MSC-graphs were already mentioned in Chapter 2. For a survey of recent results on verification and other techniques of analysis on MSC we reffer the reader to [24].

A topic less relevant to the work done in this thesis, but still relevant to the notion of race conditions in general, is the work on automatic resolving of races in MSCs. This field is, as far as we know, based mainly on the work of Bill Mitchell [20, 5, 19]. In [20, 19] the author proves, that it is possible to create, for a given MSC containing a race condition, a unique MSC without race condition (under non-fifo semantics). This result can be generalised to a proper subclass of iterative scenarios. In [9], Chen, Kalvala and Sinclair refine Mitchell's algorithm for MSC to lower the number of added messages.

Another topic closely connected to the domain of race conditions is the notion of *implementability* of a given MSC language. Intuitively, an MSC language $L$ is implementable if and only if there exist a distributed finite-state system (e.g. Communicating Finite-state Machine (CFM) – for formal definition of CFM see [14]) that accepts exactly the same language $L$.

There has been a substantial amount of work on this topics and several possible implementation models have been proposed. Although all work is based on some kind of CFM model, aproaches differ with respect to the definition of message passing (synchronous vs. asynchronous), determinism or nondeterminism of the final CFM, accepting states (global vs. local), underlying semantics (e.g. FIFO, non-FIFO, global FIFO), and (diss)allowance of deadlocks or additional messages.

The seminal paper in the field of locally accepting CFM is the paper [4] from Alur, Etessami and Yannakakis, where the notion of *weak* and *safe realizability* is defined. This work has been later extended in papers [2, 7, 18, 21].

A similar amout of work has been achieved for globally accepting CFM – the first paper [22] from Mukund, Kumar and Sohoni proposed an algorithm for implementation of arbitrary regular MSC language into a deterministic CFM. This result has been improved in papers [6, 14].

## 6.2 Future Work

Based on the notion of trace race problem, proposed in this thesis, we can define a class of *trace race free* MSC languages, that might have some interesting properties.

For example, the existence of a race conditions in a system might lead to impossibility of deadlock free implementation. Due to the inherent absence of races in *trace race free* languages, we might be able to create either more effective or more general algorithm implementing the MSC language onto some CFM model.

It might be also interesting to explore the properties of trace race free languages in the domain of *learnable languages*, which is an approach taken for example by the tool SMYLE [8].

Another possible direction is to transfer the notion of *trace race conditions* onto different extensions of MSC formalism – Compositional Message Sequence Charts (see [17]), Causal Message Sequence Charts ([13]) etc.

**Chapter 7**

# Conclusions

In this thesis, we propose a modified definition of race condition problem called the *trace race conditions*. The core idea is to apprehend different MSCs created from runs in the MSC-graph/HMSC as representations of logically distinct behaviours. In other words, all possible ambiguities of a behaviour in a system must be dealt with within the run, that specifies it.

When compared to the original race condition problem, which is undecidable, we have proved the *trace race condition* problem to be decidable for arbitrary MSC-graph/HMSC even with coregions and connections. We have also shown the problem to be NP-complete.

This thesis is based on a joint paper [12]. My main contribution to the presented results is the creation of the deciding algorithm itself as well as the proof of its correctness.

## Bibliography


[1] A. Muscholl and D. Peled. Analyzing Message Sequence Charts. In *SAM 2000: Proceedings of the 2nd Conference on MSC and SDL*, pages 3–17, Grenoble, 2000. VERIMAG, IRISA, SDL Forum.

[2] R. Alur, K. Etessami, and M. Yannakakis. Realizability and verification of MSC graphs. *Theoretical Computer Science*, 331(1):97–114, 2005.

[3] R. Alur, G.J. Holzmann, and D. Peled. An Analyzer for Message Sequence Charts. In *TACAS'96*, LNCS, pages 35–48. Springer, 1996.

[4] Rajeev Alur, Kousha Etessami, and Mihalis Yannakakis. Inference of message sequence charts. In *International Conference on Software Engineering*, pages 304–313, 2000.

[5] P. Baker, P. Bristow, C. Jervis, D. King, R. Thomson, B. Mitchell, and S. Burton. Detecting and resolving semantic pathologies in UML sequence diagrams. In *Proceedings of the 10th European software engineering conference held jointly with 13th ACM SIGSOFT international symposium on Foundations of software engineering*, pages 50–59. ACM press, 2005.

[6] N. Baudru and R. Morin. Safe implementability of regular message sequence chart specifications. In *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, pages 210–217, 2003.

[7] N. Baudru and R. Morin. Synthesis of Safe Message-Passing Systems. In *FSTTCS 2007: Foundations of Software Technology and Theoretical Computer Science*, volume 4855 of *LNCS*, pages 277–289. Springer, 2007.

[8] B. Bollig, J.P. Katoen, C. Kern, and M. Leucker. Replaying Play in and Play out: Synthesis of Design Models from Scenarios by Learning. In *Proceedings of the 13 thInternational Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS'07)*, volume 4424 of *LNCS*, pages 435–450. Springer, 2007.

[9] C.A. Chen, S. Kalvala, and J. Sinclair. Race Conditions in Message Sequence Charts. In *APLAS 2005: Programming Languages and Systems*, volume 3780 of *LNCS*, pages 195–211. Springer, 2005.

[10] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971.

[11] E. Elkind, B. Genest, and D. Peled. Detecting Races in Ensembles of Message Sequence Charts. In *TACAS'07*, volume 4424 of *LNCS*, pages 420–434. Springer, 2007.

[12] J. Fousek, V. Řehák, and P. Slovák. Decidable Race Conditions in High-Level Message Sequence Charts. In *Proceeding of International Conference on Concurrency Theory (CONCUR'08)*, LNCS. Springer, 2008. Submited for publication.

[13] T. Gazagnaire, B. Genest, L. Helouet, PS Thiagarajan, and S. Yang. Causal Message Sequence Charts. In *CONCUR 2007 : Concurrency Theory*, volume 4703 of *LNCS*, pages 166–180. Springer, 2007.

[14] B. Genest, A. Muscholl, and D. Kuske. A Kleene theorem for a class of communicating automata with effective algorithms. In *DLT : Developments in Language Theory*, volume 3340 of *LNCS*, pages 30–48. Springer, 2004.

[15] ITU Telecommunication Standardization Sector Study group 10. ITU recommandation Z.120, Message Sequence Charts (MSC), 1996.

[16] ITU Telecommunication Standardization Sector Study group X. ITU recommandation Z.120, Message Sequence Charts (MSC), 1993.

[17] E.L. Gunter, A. Muscholl, and D. Peled. Compositional message sequence charts. *International Journal on Software Tools for Technology Transfer (STTT)*, 5(1):78–89, 2003.

[18] M. Lohrey. Realizability of high-level message sequence charts: closing the gaps. *Theoretical Computer Science*, 309(1-3):529–554, 2003.

[19] B. Mitchell. Inherent Causal Orderings of Partial Order Scenarios. In *ICTAC 2004 : Theoretical Aspects of Computing*, volume 3407 of *LNCS*, pages 113–127. Springer, 2005.

[20] B. Mitchell. Resolving Race Conditions in Asynchronous Partial Order Scenarios. *IEEE Transactions on Software Engineering*, 31(9):767–784, 2005.

[21] A. Mousavi, B. Far, and A. Eberlein. The Problematic Property of Choice Nodes in high-level Message Sequence Charts. Technical report, Technical report, Laboratory for Agent-Based Software Engineering, University of Calgary, 2006.

[22] M. Mukund, K.N. Kumar, and M. Sohoni. Synthesizing distributed finite-state systems from MSCs. In *Proceedings of the 11th International Conference on Concurrency Theory (CONCUR 2000)*, volume 1877 of *LNCS*, pages 521–535. Springer, 2000.

[23] A. Muscholl and D. Peled. Message Sequence Graphs and Decision Problems on Mazurkiewicz Traces. In *MFCS'99*, volume 1672 of *LNCS*, pages 81–91. Springer, 1999.

[24] A. Muscholl and D. Peled. Deciding Properties of Message Sequence Charts. In *Scenarios: Models, Transformations and Tools*, volume 3466, pages 43–65. Springer, 2005.